Humanizing Facets Design

Goal

Provide a mechanism for improving the data in the CMR facets that will address issues found in recent Earthdata Search Client user studies and improve discovery of Earth Science data.

Traceability

⚠ CMR-2994 - JIRA project doesn't exist or you don't have permission to view it.

Issues in Current Facets

User studies with the Earthdata Search Client has shown there are issues with the facets the CMR produces.

- Misspellings ("Bioosphere" instead of Biosphere)
- Older terms (AM-1) alongside a newer term (Terra) that refer to the same thing.
- Different names for the same thing. (Processing "Level 1" and "1")
- Facets that don't provide much value (Platform with the name "SATELLITE")
- The most important facet terms aren't presented at the top.
- Facets in all UPPER CASE which is slower for most people to read. "Most readers spend the bulk of their time reading lowercase text and are therefore more proficient at it."
  - Citation https://www.microsoft.com/typography/ctfonts/wordrecognition.aspx
- Extra whitespace around some strings.

The CMR will provide a new mechanism to correct these problems and humanize the facets terms for human facing tools like EDSC. The humanization will be done by creating various transformation instructions via the Ingest API. The Indexer will apply these transformations to allow searching and faceting by a new set of humanized fields. The transformations will not be applied to the existing fields.

Example Workflow

This section describes how the transformations are created, used with indexing, and ultimately returned from the CMR to present to the user.

Create Humanizer via Ingest API

Humanizers are transformation instructions which are created via Ingest. The following request creates a humanizer that says to replace platform short name of "AM-1" with "Terra". The priority field is used to indicate the order of application for transformations. (More on that later)

Request:

```
POST /ingest/humanizer
{"type": "alias",
"field": "platform",
"source_value": "AM-1",
"replacement_value": "Terra",
"priority": 10}
```

Ingest stores the humanizer using Metadata DB and returns the concept id and revision id in the response.

```
HTTP 200
{"concept_id":"H1-CMR", "revision_id":1}
```

Collections Indexed with Humanized Fields

The Indexer will keep the humanizer records locally in a cache using the consistent cache to keep the copies the same on all instances. It will listen for humanizer ingest events sent from Metadata DB as a sign that it should update it's cache.

Indexing currently follows the following process:

1. Metadata DB concept saved and message broadcast.
2. Indexer receives broadcast message.

3. Indexer retrieves concept from Metadata DB
4. Indexer parses concept into UMM records.
5. Indexer converts UMM record into elastic document.
6. Indexer sends elastic document to ElasticSearch.

The process will be updated to include the application of humanizer transforms.

1. Metadata DB concept saved and message broadcast.
2. Indexer receives broadcast message.
3. Indexer retrieves concept from Metadata DB
4. Indexer parses concept into UMM records.
5. ***Indexer applies humanizer transforms to produce UMM record with additional humanized fields.***
6. Indexer converts UMM record into elastic document ***including humanized fields*****.**
7. Indexer sends elastic document to ElasticSearch.

Example UMM Records

A subset of a UMM record prior to transformation.
```
{:Platforms [{:ShortName "AM-1"
...}]
...}
```

After the change the record looks like the following. Notice that there's a new namespaced key within the existing record. We'll use namespaced keywords to keep them separate.
```
{:Platforms [{:ShortName "AM-1"
:cmr.humanized/ShortName "Terra"
...}]
...}
```

Example Indexed ElasticSearch Document

This is an example of the ElasticSearch document that would be sent.

```
{:platform-sn "AM-1"
:platform-sn.lowercase "am-1"
:humanized-platform-sn "Terra"}
:humanized-platform-sn.lowercase "terra"
...}
```

Reindexing Collections on Humanizer changes.

Collections will need to be reindexed when a humanizer changes. The initial plan is to manually reindex all collections after a set of humanizer changes has been applied. Eventually we can make this process automated where the collections that relate to the humanizer change are found and queued for reindexing.

Exposing Humanized Facets through Search

Search will expose the humanized facets via the Updated Facet Response (https://wiki.earthdata.nasa.gov/display/CMR/Updated+facet+response). The user will see "Terra" in the EDSC. Selecting it will trigger a search using the humanized parameter name `humanized_platform`.

APIs

Ingest

- /ingest/humanizers
    - POST - Create a humanizer
    - GET - Retrieve all humanizers
    - /:concept-id (H1234-CMR)
        - PUT - Update a humanizer
        - GET - Retrieve a humanizer
        - DELETE - Delete a humanizer.

Humanizer Representations

A humanizer is represented as a JSON map with a type field. The other fields that are present are determined by the type.

Fields Present in every Humanizer

- **type**
    - Indicates what type of humanizer this is.
    - One of
        - alias
        - trim_whitespace
        - ignore
        - capitalize
        - set_facet_priority_order
- **field** - Identifies the field
    - platform, instrument, science_keyword, project, processing_level, data_center
- **source_value** - (optional, default: none) If supplied, only fields with the given values will be updated
- **order** - (optional, default: 0) Used to determine the order of application of the humanizers.

Alias Humanizer

Humanizes a facet value by replacing an existing matching value with a different value. ie. "AM-1" -> "Terra"

- replacement_value - The replacement value to use.

Trim Whitespace Humanizer

Removes trailing and ending whitespace.

Ignore Humanizer

Skips over a facet value that is not applicable to the user. Example: the "SATELLITE" platform.

Capitalize Humanizer

Changes the case of strings found to Capital Case like you would refer to a multiword proper noun.

Examples:

- "INPUT" -> "Output"
- "MULTI_WORD" -> "Multi Word"
- "MULTI-WORD" -> "Multi Word"
- "multi word" -> "Multi Word"

Set Facet Priority Order Humanizer

Sets the order in which that the value should be returned. Normally facets are returned from the CMR in order of count. The count indicates a level of relevancy based on how many collections reference that value. This humanizer would allow manually indicating the relevancy of this value.

- priority - A number indicating the priority (default: 0)

Detailed Example:

There are 5 different platforms in a fictional CMR instance: Alpha, Bravo, Charlie, Delta, and Eagle.

The numbers of collections that reference each value is the following.

- Charlie - 50
- Delta - 25
- Bravo - 27
- Eagle - 10
- Alpha - 2

Priorities have been set on some of these facets. Eagle and Bravo are very important to users so they have the following priority values. The other platforms are not as important.

- Eagle - Priority 100
- Bravo - Priority 50

A request is sent to the CMR for facets and the number of facets requested is set to 4. The facets would be requested from Elasticsearch sorted by priority descending and then count descending and limited to the top 4. Elasticsearch would return the following:

- Eagle (Priority 100, Count 10)
- Bravo (Priority 50, Count 27)
- Charlie (Priority 0, Count 50)
- Delta (Priority 0, Count 25)

The CMR would return these to the client. EDSC would then sort them alphabetically and display to the end user. The end user would see each facet value with the count

- Bravo - 27
- Charlie - 50
- Delta - 25
- Eagle - 10

Enforcing ACLs for Modifying Humanizers

We should add a new System level target for CRUD of humanizers. We will wait for the new Access Control Service support of ACLs to be finished. Until that time we'll use an existing system target that's not otherwise used. (See http://api.echo.nasa.gov/echo/ws/v10/SystemObject.html) We suggest the SYSTEM_INITIALIZER system target. We'll only need to grant it to the EDSC user and the administrators.

Stories

- Ingest CRUD alias command stories
  - As a Client User, I can create a humanizer.
  - As a Client User, I can update a humanizer.
  - As a Client User, I can retrieve a humanizer.
  - As a Client User, I can retrieve all humanizers.
  - As a Client User, I can delete a humanizer.
  - As a Client User, I want collections to be automatically reindexed when a humanizer change occurs that would impact how this collection is indexed.
    - Not an MVP story
  - As a Client User, I want the CMR to reject manipulation of humanizers if I do not have the appropriate humanizer permissions.
    - Not MVP. See section above on ACLs. This will be a new system level HUMANIZERS target that we'll add once ACLs are in place in access control service.
- Search Stories
  - As a Client User, I can search for collections matching a value within a humanized field using a parameter search.
  - As a Client User, I can search for collections matching a value within a humanized field using a JSON Query.
- Facet Stories
  - As a Client User, I would like facets sorted by priority specified via humanizers.
- Tasks
  - Metadata DB task to save humanizers
  - Implement humanizers in the UMM Spec library
  - Apply humanizers during indexing
  - Add humanizer cache to the indexer

Error rendering macro 'pageapproval' : null